Machine Learning Techniques to Predict Software Defect

Ramakanta Mohanty
Keshav Memorial Institute of Technology, India

Vadlamani Ravi
Institute for Development and Research in Banking Technology (IDRBT), India

INTRODUCTION

Machine learning techniques have been dominating in the last two decades. The recently published comprehensive state-of-the-art review (Mohanty et al., 2010) justifies this issue. The ability of software quality models to accurately identify critical faulty components allows for the application of focused verification activities ranging from manual inspection to automated formal analysis methods. Therefore, software quality models to ensure the reliability of the delivered products. Accurate prediction of fault prone modules enables the verification and validation activities that includes quality models: Musa, 1998, logistic regression (Basili et al., 1996), discriminant analysis (Khoshgoftaar, 1996), the discriminant power techniques (Schneidewind, 1992), artificial neural network (Khoshgoftaar, 1995), genetic algorithm (Azar et al., 2002), and classification trees (Gokhale et al., 1997; Khoshgoftar et al., 2002; Selby et al., 1988; Fenton et al., 1999).

A wide range of modeling techniques has been proposed and applied for software quality predictions. These include: proposed the Bayesian belief network as the most effective model to predict software quality.

Classification is a popular approach to predict software defects and involves categorizing modules, which is represented by a set of metrics or code attributes into fault prone (fp) non fault prone (nfp) by means of a classification model derived from data (Lessman et al., 2008), statistical methods (Basili et al., 1996; Khoshgoftar & Allen, 1999), tree based methods, (Guo et al., 2004; Khoshgoftar et al., 2000; Menzies et al., 2004; Porter et al., 1990; Selby et al., 1988), neural networks (Khoshgoftar et al., 1995, 1997) and analogy based approaches (El-Emam et al., 2001; Ganeshan et al., 2000; Khoshgoftar et al., 2003), Decision tree (Selby et al., 1988). The discriminative power techniques correctly classified 75 out of 81 fault free modules, and 21 out of 31 faulty modules (Porter et al., 1992). Lessmann et al., (2008) used 10 software development datasets from NASA MDP repository to predict software defects. Most recently, Pendharkar (2010) used the same dataset to test the efficacy of their hybrid exhaustive search and probabilistic neural network (PNN), and simulated annealing (SA) method.

In this chapter, we present a software defect prediction methodology based on GP, BPNN, GMDH, PNN, GRNN, TreeNet, CART, Random Forest Naïve Baye's and J48 on the DATATRIEVE, PC1, PC3, PC4, MC1, KC1, KC2, KC3, CM1 and JM1 datasets.

The rest of the chapter is organized in the following manner. A brief discussion about the overview of machine learning techniques is presented in section 2. Section 3 describes the experimental methodology. Section 4 presents a detailed discussion of the results and discussions. Finally, section 5 concludes the chapter.

DOI: 10.4018/978-1-4666-5202-6.ch129

OVERVIEW OF THE TECHNIQUES APPLIED

Here we present a brief overview of the machine learning, soft computing and statistical techniques that are employed in this chapter. Since, BPNN is too popular to be overviewed here, the rest of the techniques are presented here.

Group Method of Data Handling (GMDH)

The GMDH was proposed by Ivakhnenko (1968). The main idea behind GMDH is that it tries to build a function (called a polynomial model) that would behave in such a way that the predicted value of the output would be as close as possible to the actual value of output (http://www.inf.kiew.ua/gmdhhome).GMDH (Farlow, 1984) is a heuristic self organizing method that models the input-output relationship of a complex system modeling.

GMDH model with multiple inputs and one output is a subset of the components of the base function in Equation (1) as

$$Y\left(x\_1,\ x\_(2)...,x\_(n)\right)= a_0 + \sum_{i=1}^{m} a_i f_i \tag{1}$$

where f is an elementary function depends on different sets of inputs, a_i represents coefficients and m represent the number of base function components. In order to find the best solution GMDH algorithm considers various component subsets of the base function called partial models. The coefficients of these models are estimated by the least squares model.

Genetic Programming

GP is a search methodology that starts from a high-level statement of 'what needs to be done' and automatically creates computer programs to solve the problem. This population of programs is progressively evolved over a series of generations (Poli, 2008; Koza, 1992). GP randomly generates an initial population of solutions. The initial population is manipulated using various genetic operators to produce new populations. These operators include reproduction crossover, mutation. We used the GP implementation available at http://www.rmltech.com.

J48 (Weka)

J48 algorithm was developed by J. Ross Quilan, the very popular C4.5. Decision trees are a classic way to represent information from machine learning and offer a fast way to express structures in data.

CART

CART was introduced by Breiman et al. (1984) can solve both classification and regression problems (http://salford-systems.com). Decision tree algorithms induce a binary tree on a given training data, resulting in a set of 'if–then' rules. These rules can be used to solve the classification or regression problem. The key elements of a CART analysis (1984) are a set of rules for: (i) splitting each node in a tree, (ii) deciding when a tree is complete; and (iii) assigning each terminal node to a class outcome (or predicted value for regression). We used the CART implementation available at http://salford-systems.com.

TreeNet

TreeNet was introduced by Friedman (1999). It makes use of a new concept of 'ultra slow learning' in which layers of information are gradually peeled off to reveal structure in data. TreeNet models are typically composed of hundreds of small trees, each of which contributes just a tiny adjustment to the overall model. TreeNet is insensitive to data errors and needs no time-consuming data reprocessing or imputation of missing values.